





Using XML in .NET

Agenda: Day 1

- Module 01: What is Microsoft® .NET?
- Module 02: Visual Studio® .NET
- Module 03: The .NET Languages
- Module 04: Building ASP® .NET Applications
- Module 05: Using Data Access in .NET Applications
- Module 06: Object Oriented Concepts in .NET
- Module 07: Creating Windows Applications
- Module 08: Using XML in .NET

Module 08: Agenda

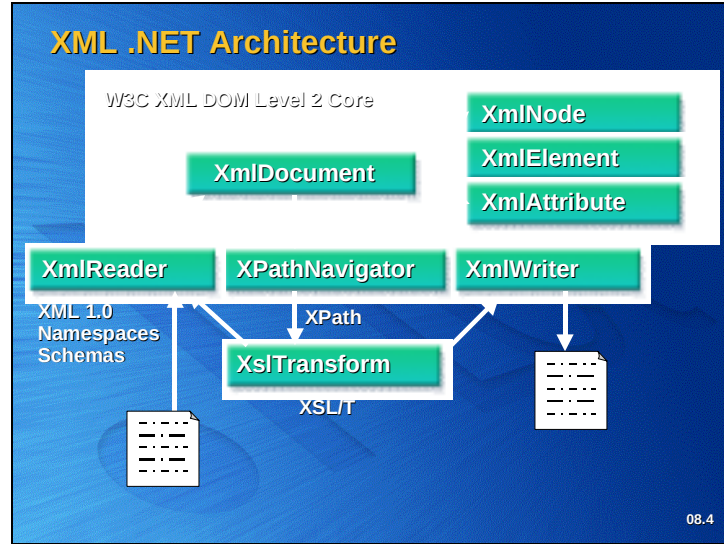
- Parsing XML
 - XmlReader and XmlWriter
 - XML document
- Navigating XML
 - XPathNavigator
- Data and XML
 - Managed Data Overview
 - ADO.NET and XML
 - Accessing XML documents as relational data
 - Accessing relational data as XML

08.3

Module 08 Agenda

What you will learn:

- How to work with XML in .NET
- How to access XML data as relational data
- How to access relational data as XML



XML .NET Architecture

The XML Framework in .NET provides a comprehensive and integrated set of classes, so that you can work with XML documents and data. The XML classes in the .NET Framework can be broken into several groups: Parsing and writing XML, validating XML, editing an XML document by using **XmlReader**, **XmlWriter**, and the **XmlDocument**. The **XPath**, **XsltTransform**, and **XsltSchema** classes support applying XPath queries, performing XSL Transformations (XSLT) and editing the XML Schema Definition (XSD) language schema.

The **XmlDocument** class implements the W3C Document Object Model (DOM) Level 1 Core and the DOM Level 2 Core. The DOM is an in-memory (cache) tree representation of an XML document and enables the navigation and editing of this document. Because **XmlDocument** implements the **IXPathNavigable** interface, it can also be used as the source document for the **XsltTransform** class.

The **System.Xml.XmlDataDocument** class extends **XmlDocument** and allows structured data to be stored, retrieved, and manipulated through a relational **System.Data.DataSet**. This class allows components to mix XML and relational views of the underlying data.

XmlReader and XmlWriter

- **XmlReader**
 - Pull model interface
 - Cursor over forward only stream
 - Parse time validation
 - XmlTextReader, XmlNodeReader Implementations
- **XmlWriter**
 - Helps generate well formed XML output
 - Helps with namespaces
 - XmlTextWriter implementation

08.5

XmlReader and XmlWriter

At the core of the .NET Framework XML classes are two abstract classes: XmlReader and XmlWriter.

The **XmlReader** defines methods that enable you to pull data from XML or to skip unwanted records

XmlReader provides a fast, forward-only and read-only cursor for processing an XML document stream.

XmlWriter provides an interface for producing XML document streams that conform to the W3C's XML 1.0 + namespace recommendations. Applications that want to process XML documents use XmlReader, while applications that want to produce XML documents use XmlWriter. Both classes imply a streaming model that doesn't require an expensive in-memory cache. This makes them both attractive alternatives to the classic DOM approach.

Both XmlReader and XmlWriter are base classes that are abstract that define the functionality that all derived classes must support. There are three concrete implementations of XmlReader:

- XmlTextReader
- XmlNodeReader and
- XslReader

There are two concrete implementations of XmlWriter

- XmlTextWriter
- XmlNodeWriter

XmlTextReader and XmlTextWriter support, reading from and writing to a text-based stream, while XmlNodeReader and XmlNodeWriter are designed to work for in-memory DOM trees.

One of the biggest advantages of this new design is that custom readers and writers can be developed to extend the built-in functionality.



XmlReader

Demo

XmlReader

The purpose of this demo is to show reading XML from URL:

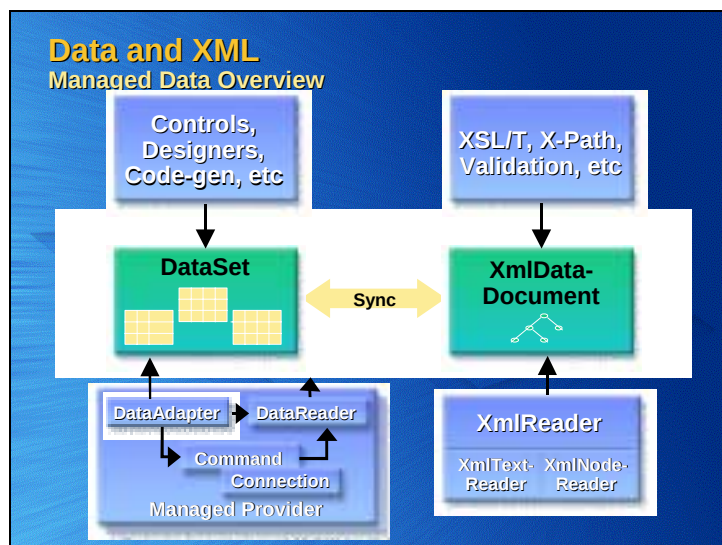
- How to use the XmlReader?
- How to load XML from a URL?
- How to extract elements from the resulting document?

XMLDocument

- Additional functionality – Load(), Save() and event handling
- InnerXML properties build a subtree for you
`node.InnerXML = "<Person><Name>Mark</Name><Email>mfussell@microsoft.com</Email></Person>"`

08.7

XmlDocument



Data and XML

Managed Data Overview

This graphic shows the relationship between the DataSet and the XmlDocument and the Objects that act on each. The synchronization process between the two exists because they are managed data.

The Common Language Runtime automatically handles object layout and manages references to objects, releasing them when they are no longer being used. Objects whose lifetimes are managed in this way are called **managed data**. Automatic memory management eliminates memory leaks as well as some other common programming errors. If your code is managed, you can use managed data, unmanaged data, or both managed and unmanaged data in your .NET application. Because language compilers supply their own types, such as primitive types, you might not always know (or need to know) whether your data is being managed.

ADO.NET and XML

- Natural evolution of ADO and MSXML
- Fully integrated
- DataSet and XmlDataDocument are key components
- An XML data set can be presented as an XML document and vice versa

08.9

ADO.NET and XML Framework

- XML Framework provides easy and extensible classes
- Unified and consistent data and XML architecture
- Databinding

ADO.NET and XML

- **DataSet**
 - Loads/saves XML data into/out of DataSet
 - Schema can be loaded/saved as XSD
 - Schema can be inferred from XML Data
- **DataSet with XmlDataDocument**
 - Exposes relational view over structured XML
 - According to the DataSet Schema
 - Allows strong typing, control binding and relational access of XML data
 - Allows XML tools (schema validation, XSL/T, XPath queries) against relational data
 - Preserves full fidelity of XML document

08.10

ADO.NET and XML

ADO.NET and the .NET XML framework converge in the **DataSet** object. The **DataSet** can be populated with data from an XML source, whether it is a file or an XML stream. The **DataSet** can be written as W3C compliant XML, including its schema as XML Schema Definition language (XSD) schema, regardless of the source of the data in the **DataSet**. Because the native serialization format of the **DataSet** is XML, it is an excellent medium for moving data between tiers making the **DataSet** an optimal choice for remoting data and schema context to and from a Web Service.

The **DataSet** can also be synchronized with an **XmlDataDocument** to provide relational and hierarchical access to data in real time.

The ADO.NET **DataSet** provides you with a relational representation of data. For hierarchical data access, you can use the XML classes available in the .NET Framework. Historically, these two representations of data have been used separately. However, the .NET Framework enables real-time, synchronous access to both the relational and hierarchical representations of data through the **DataSet** object and the **XmlDataDocument** object, respectively.

The relationship between the **DataSet** and the **XmlDataDocument** creates great flexibility by allowing a single application, using a single set of data, to access the entire suite of services built around the **DataSet** (such as Web Forms and Windows Forms controls, and Visual Studio.NET designers), as well as the suite of XML services including Extensible Stylesheet Language (XSL), XSL Transformations (**XSLT**), and XML Path Language (**XPath**). You do not have to choose which set of services to target with the application—both are available.

DataSet with XML Sources

- Load DataSet with XML
 - ReadXML method (and more)
 - Open as relational data
- Saving DataSet as XML
 - WriteXML method

08.11

DataSet with XML Sources

You also can do x/path, validation and transformations.

When XML is used to load a DataSet, the data becomes relational and loses some of its XML characteristics:

The process only preserves what is meaningful to relational view.

- Element ordering may change
- White space is not preserved
- Content not mapped to DataSet Schema is discarded

To preserve fidelity of XML document, use the XmlDocument

Load DataSet with XML

```
Dim ds As new DataSet()  
ds.ReadXML ("inventory.XML")
```

Add a record to the inventory table

```
Dim inventory as DataTable = ds.Tables("Inventory")  
Dim row as DataRow = inventory.NewRow()  
Row ("TitleID") = 1  
Row ("Quantity") = 25  
inventory.Rows.Add(row)
```

Write out XML

```
ds.WriteXML ("inventory.XML")
```

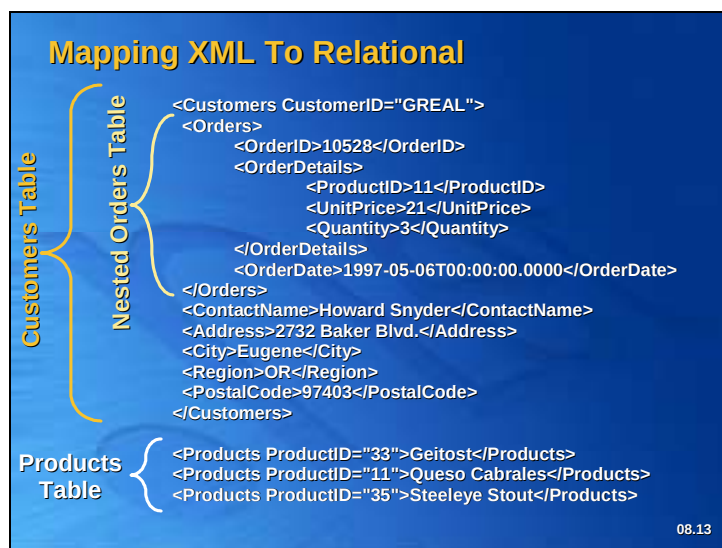
Schema Inference

- Schema can be inferred from data
 - When no schema is defined when you call `DataSet.ReadXML()`
- General rules
 - Element becomes table if:
 - It repeats or contains more than simple content
 - Otherwise, it becomes column
 - Attributes become columns
 - Relations are created for nested table-mapped elements
- Useful for ad hoc binding to data
 - Schema depends on data instance

08.12

Schema Inference

Schema inference is a two-step process. First, the **DataSet** extracts elements that would be defined as tables according to the rules that follow. Then the remaining elements are defined columns in one of the defined tables.



Mapping XML to Relational

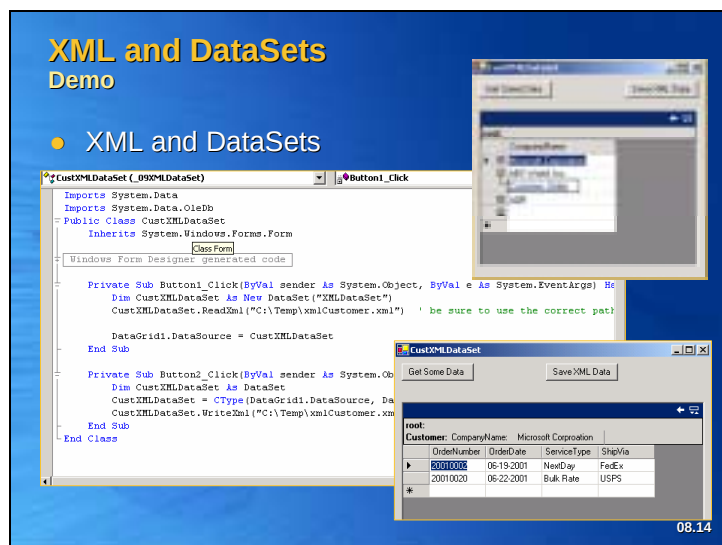
Inference of schema depends upon instances of data. For example, the following two XML streams infer different schemas:

```
<Orders>
  <OrderID>10528</OrderID>
  <OrderDetails>
    <ProductID>Geitost</ProductID>
    <UnitPrice>21</UnitPrice>
    <Quantity>3</Quantity>
  </OrderDetails>
</Orders>
```

In this case, Orders table has ProductID" as a Column

```
<Orders>
  <OrderID>10528</OrderID>
  <OrderDetails>
    <ProductID>Geitost</ProductID>
    <ProductID>Queso Cabrales</ProductID>
    <UnitPrice>21</UnitPrice>
    <Quantity>3</Quantity>
  </OrderDetails>
</Orders>
```

In this case, Orders table has a relation to the ProductID table.



XML and DataSets

Demo

XML and DataSets

- Show how to load and save XML using datasets
- Sample of working with DataSet and XML, Load dataset from XML, bind to datagrid

Demonstrate:

- Loading a dataset from XML
- Databinding to the dataset
- Saving changes from the grid back to the XML document

XML Relational Data

- XmlDataDocument
 - SQL XML
 - XML streamed from SQL Server™
 - Support for UpdateGrams

08.15

XML Relational Data

Execute SQL and load XmlReader

```
Dim cmd as SqlCommand = cnn.CreateCommand()  
cmd.CommandText = "Select * from authors FOR XML"  
Dim xr as XmlReader = cmd.ExecuteXmlReader()  
Read Data Directly  
While xr.Read  
    ...  
End While  
Load data into DataSet  
Dim ds as New DataSet()  
ds.ReadXml(xr, XmlReadMode.SqlXml)
```

XmlDataDocument

- A DataSet can be associated with an XmlDocument
 - Exposes a relational view over structured XML
 - Preserves full fidelity of XML document
 - Simultaneously exposes data relationally or as XML
 - Semi-structured data
 - Multiple tools on same data synchronized hierarchical/relational view
- Mapping is based on W3C XSD schemas
- Provides efficient connection of relational data to XslTransform

08.16

XmlDataDocument

The **XmlDataDocument**, a subclass of the DOM in this namespace, provides relational access to data and maps XML to relational data in an ADO.NET **DataSet**. Microsoft .NET-based Applications can use the classes in the **System.XML** namespace to access and manipulate both XML documents and relational data in the **XmlDataDocument**. This implementation supports n-tiered architectures for collecting and distributing data.

```
// Associate an XmlDocument with the DataSet
XmlDataDocument doc = new XmlDocument(ds1);
XPathNavigator nav = doc.CreateNavigator();
// Create an iterator to iterate all the records in Customers DataTable
XPathNodeIterator iterator = nav.Select("// Customers[*]");
// Write out the Customer Details
while(iterator.MoveNext())
{
    Console.WriteLine(iterator.Current.Value);
}
```

Summary

- .NET and XML
 - Objects and features make working with XML easy
 - XML is used under the hood by many objects (such as the DataSet)
 - DataSet provides relational view
 - Persists/loads data as XML
 - Persists/loads schema as XSD
 - Serializes as XML with in-line schema
 - XmlDataDocument provides an XML view
 - Exposes relational data to XML tools
 - Exposes a relational subset of XML data
 - Preserves fidelity of XML

08.17

Summary

Resources

- The Extensible Markup Language (XML) is the universal format for structured documents and data on the Web; for more information: <http://www.w3.org/xml/>

08.18

Resources

The Extensible Markup Language (XML) is the universal format for structured documents and data on the Web; for more information: <http://www.w3.org/xml/>

In-Depth Training on XML

Direct from Microsoft

- 2500A: Introduction to XML and the Microsoft .NET Platform
- 2503A: Programming with XML in the Microsoft .NET Framework
 - Availability Date to be Decided

08.19

In-Depth Training on XML

Direct from Microsoft

